# Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-hoc Networks

Sheng Zhong, Yang Richard Yang, Jiang Chen
Department of Computer Science, Yale University, New Haven, CT 06520
{sheng.zhong,yang.r.yang,jiang.chen}@yale.edu

***Abstract*— Mobile ad hoc networking has been an active research area for several years. How to stimulate cooperation among selfish mobile nodes, however, is not well addressed yet. In this paper, we propose Sprite, a simple, cheat-proof, credit-based system for stimulating cooperation among selfish nodes in mobile ad hoc networks. Our system provides incentive for mobile nodes to cooperate and report actions honestly. Compared with previous approaches, our system does not require any tamper-proof hardware at any node. Furthermore, we present a formal model of our system and prove its properties. Evaluations of a prototype implementation show that the overhead of our system is small. Simulations and analysis show that mobile nodes can cooperate and forward each other's messages, unless the resource of each node is extremely low.**

***Index Terms*— Mobile Ad Hoc Networks, Network Security, Payment Scheme, Game Theory, Incentive.**

## I. INTRODUCTION

IN recent years, mobile ad hoc networks have received much attention due to their potential applications and the proliferation of mobile devices [25], [32]. Specifically, mobile ad hoc networks refer to wireless multi-hop networks formed by a set of mobile nodes without relying on a preexisting infrastructure. In order to make an ad hoc network functional, the nodes are assumed to follow a self-organizing protocol, and the intermediate nodes are expected to relay messages between two distant nodes. Recent evaluations have shown that ad hoc networks not only are flexible and robust, but also can have good performance in terms of throughput, delay and power efficiency [14].

So far, applications of mobile ad hoc networks have been envisioned mainly for emergency and military situations. In such applications, all the nodes of the network belong to a single authority and therefore have a common objective. As a result, cooperation among the nodes can be assumed. However, as observed by several authors [20], [4], [5], [2], [3], it may soon be possible to deploy ad hoc networks for civilian applications as well. In such emerging civilian applications, the nodes typically do not belong to a single authority. Consequently, cooperation behaviors such as forwarding each other's messages, cannot be directly assumed.

We can identify two types of uncooperative nodes: faulty/malicious nodes and selfish nodes. By saying faulty/malicious nodes, we refer to the broad class of nodes that are either faulty and therefore cannot follow a protocol, or are intentionally malicious and try to attack the system. The problems of faulty/malicious nodes need to be addressed from many layers, for example, using spread-spectrum encoding to avoid interference over the communication channel; using a reputation system to identify the faulty/malicious nodes and subsequently avoid or penalize such nodes [20]; and applying the techniques from fault-tolerant computing to perform computation correctly even in the presence of faulty/malicious nodes. Although the problems of faulty/malicious nodes can be important in military applications, the focus of this paper is on selfish nodes, which we expect will be the dominant type of nodes in a civilian ad hoc network.[1] Specifically, a selfish node is an economically rational node whose objective is to maximize its own welfare, which is defined as the benefit of its actions minus the cost of its actions. Since forwarding a message will incur a cost (of energy and other resources) to a node, a selfish node will need incentive in order to forward others' messages.

One possibility to provide incentive is to use a reputation system [20], [2], [3]. For example, in [20], Marti et al. proposed a reputation system for ad hoc networks. In their system, a node monitors the transmission of a neighbor to make sure that the neighbor forwards others' traffic. If the neighbor does not forward others' traffic, it is considered as uncooperative, and this uncooperative rep-

Research sponsored in part by NSF grant no. ANIR–0207399.

[1] Note that a complete system can include both a component to deal with faulty/malicious nodes and a component to provide incentive to selfish nodes, using the technique proposed in this paper.

utation is propagated throughout the network. In essence, we can consider such a reputation system as a repeated game whose objective is to stimulate cooperation (*e.g.*, see Chapter 8 of [23]). Such reputation systems, however, may have several issues. First, there is no formal specification and analysis of the type of incentive provided by such systems. Second, these systems have not considered the possibility that even selfish nodes can collude with each other in order to maximize their welfare. Third, some of the current systems depend on the broadcast nature of wireless networks in order to monitor other nodes. Such monitoring, however, may not always be possible due to asymmetric links when nodes use power control. Furthermore, directional antennas [29], [33], which are gaining momentum in wireless networks in order to improve capacity, will also make monitoring hard.

Another possibility to provide incentive is to use credit (or virtual currency). Buttyan and Hubaux proposed a nice solution of this type in [4], and then presented an improved result based on credit counters in [5]. For both proposals, a node receives one unit of credit for forwarding a message of another node, and such credits are deducted from the sender (or the destination). Besides other potential issues that we will discuss in Section II, both proposals require a tamper-proof hardware at each node so that the correct amount of credit is added or deducted from the node. As a result of this requirement, although both proposals are interesting, they may not find widespread acceptance.

In this paper, we propose Sprite, a simple, cheat-proof, credit-based system for mobile ad-hoc networks with selfish nodes. Similar to [4] and [5], our system also uses credit to provide incentive to selfish nodes. However, one of the novel and distinguishing features is that our system does not need any tamper-proof hardware at any node.

At a high level, the basic scheme of our system can be described as follows. When a node receives a message, the node keeps a *receipt* of the message. Later, when the node has a fast connection to our Credit Clearance Service (CCS), it reports to the CCS the messages that it has received/forwarded by uploading its receipts. The CCS then determines the charge and credit to each node involved in the transmission of a message, depending on all the reported receipts of a message.

There are two issues regarding the design of our system. First, since there is no tamper-proof hardware at any node and the charge and credit are based on the reports of the selfish nodes, a selfish node (or even a group of colluding node) may attempt to cheat the system to maximize its expected welfare. As an example, a selfish node may withhold its receipt, or collude with other nodes to fake receipts, if such actions can maximize its welfare. This is the security perspective of our system. Second, a node should receive enough credit for forwarding a message for another node, so that it can send its own messages with the received credit, unless the resource of the node itself is extremely low. This is the incentive perspective of our system.

In summary, the contributions of this paper are the following. First, we present Sprite, a system to provide incentive to selfish mobile nodes to cooperate. Second, our system determines charge and credit from a game-theoretic perspective, and motivates each node to report its actions honestly, even when a collection of the selfish nodes collude. Third, we model the essential component of our system as a game and prove the correctness of our system under this model. As far as we know, this is the first pure-software solution that has formal proofs of security. Our main result works for message-forwarding in unicast, and we extend it to route discovery and multicast as well. Fourth, we perform extensive evaluations and simulations of our system. Evaluations of a prototype implementation show that the overhead of our system is small. Simulations show that the nodes will cooperate and forward each other's messages, unless the resource of each node is extremely low.

The rest of this paper is organized as follows. In Section II, we discuss related work. In Section III, we present the overall architecture and the intuitions behind our design. We then give the full specification our system in Section IV. In Section V, we present a formal model of our system and prove the security properties under this model. In Section VI, we further consider the incentive issue in route discovery and multicast. In Section VII, we present evaluations of our solution. Our conclusion and future work are in Section VIII.

## II. Related work

Three classes of work are closely related to this paper: reputation systems, two stimulation approaches from the Terminodes project, and algorithmic mechanism design.

### A. *Reputation-based approaches*

In [20], Marti et al. considered uncooperative nodes in general, including selfish and malicious nodes. In order to cope with this problem, they proposed two tools: a watchdog, which identifies misbehaving nodes, and a pathrater, which selects routes that avoid the identified nodes. Their simulations showed that these two tools can maintain the total throughput of an ad hoc network at an acceptable level even with a large percentage of misbehaving nodes.

In [2], [3], Buchegger and Le Boudec proposed and evaluated their CONFIDENT protocol, which detects and isolates misbehaving nodes. However, as we discussed in Section I, there are several issues that such reputation-based systems need to address.

### *B. Two stimulation approaches from Terminodes*

In [4], Buttyan and Hubaux proposed a stimulation approach that is based on a virtual currency, called nuglets, which are used as payments for packet forwarding. Using nuglets, the authors proposed two payment models: the Packet Purse Model and the Packet Trade Model. In the Packet Purse Model, the sender of a packet pays by loading some nuglets in the packet before sending it. Intermediate nodes acquire some nuglets from the packet when they forward it. If the packet runs out of nuglets, then it is dropped. In the Packet Trade Model, the destination of a packet pays for the packet. To implement the Packet Trade Model, each intermediate node buys a packet from its previous node for some nuglets and sells it to the next node for more nuglets. In this way each intermediate node earns some nuglets and the total cost of forwarding the packet is covered by the destination. To implement either the Packet Purse Model or the Packet Trade Model, a tamper-proof hardware is required at each node to ensure that the correct amount of nuglets is deducted or credited at each node.

Besides the requirement for a tamper-proof hardware at each node, some other issues also exist for the Packet Purse Model and the Packet Trade Model:

1) Both models require the clearance of nuglets in real-time. As a result, if the system does not have enough nuglets circulating around, the performance of their system may degrade.
2) Under both models, if a mobile node runs out of nuglets, its tamper-proof hardware still has to contact with some central authority in order to "refill" its credit. (Actually, the CSS introduced by our system is similar to such an authority.)
3) A disadvantage of the Packet Trade Model is that it is vulnerable to network overload, since the senders do not have to pay. For this reason, the authors of [4] mainly studied the Packet Purse Model.

Besides the nuglet approach, Buttyan and Hubaux also proposed a scheme based on credit counter [5]. In this new approach, each node keeps track of its remaining battery and its remaining credit. The authors simulated four rules for a node to determine when to forward others' packets and when to send its own packets. Our analysis shows that the first rule is actually optimal to achieve their given goals.[2] Although this new scheme is simple and elegant, it still requires a tamper-proof hardware at each node so that the correct amount of credit is deducted or credited. Furthermore, the first two issues we outlined in the previous paragraph exist for this approach as well.

Both [4] and [5] are the results of the Terminodes project. General reviews of the Terminodes project, and of the related security problems, can be found in [17], [31], [1], [16].

### *C. Algorithmic mechanism design and game theory*

Our approach is motivated by algorithmic mechanism design (see *e.g.*, [21], [22], [24], [13], [10], [11], [9], [27]), which is an emerging active research area in the intersection of computer science and mathematical economics. In particular, Feigenbaum et al. have considered BGP-based mechanism design for lowest-cost unicast routing in the Internet [9]. In [10], Feigenbaum et al. have considered cost sharing for multicast. Golle et al. have analyzed the incentives in peer-to-peer networks [11]. However, as far as we know, there is no previous proposed mechanism design for ad hoc networks. Furthermore, although our design is motivated by algorithmic mechanism design, our problem does not fit exactly into the mechanism-design framework. For example, in our game, the information held by each player is not totally private, while in mechanism design, each player must have a private *type*.

## III. Overview of our Approach

In this section, we present the overall architecture and the intuitions behind our design; the formal results will be presented in Sections IV and V.

### *A. System architecture*

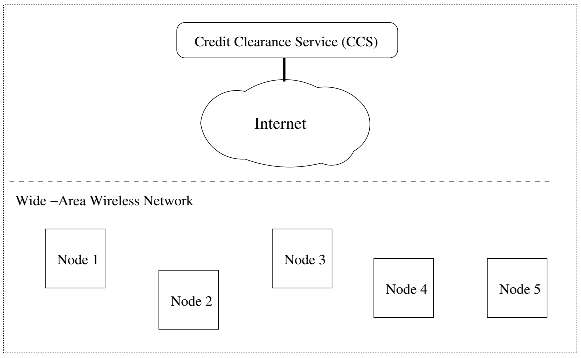


Fig. 1. The architecture of Sprite.

[2]Communicated to the authors of [5].

Figure 1 shows the overall architecture of our system, which consists of the Credit Clearance Service (CCS) and a collection of mobile nodes. The nodes are equipped with network interfaces that allow them to send and receive messages through a *wireless overlay* network [30], *e.g.*, using GPRS in a wide-area environment, while switching to 802.11 or Bluetooth in an indoor environment. To identify each node, we assume that each node has a certificate issued by a scalable certificate authority such as those proposed in [34], [19]. For concreteness of presentation, we assume that the sender knows the full path from the sender to the destination, using a secure ad hoc routing protocol based on DSR [18], [6], [15]. The incentive issues of route discovery will be investigated in Section VI.

When a node sends its own messages, the node (or the destination, see later) will lose *credit* (or virtual money) to the network because other nodes incur a cost to forward the messages. On the other hand, when a node forwards others' messages, it should gain credit and therefore be able to send its messages later.

There are two ways for a node to get more credit. First, a node can pay its debit or buy more credit with real money, at a variable rate to the virtual money, based on the current performance of the system. However, the preferred and dominant way to get more credit is by forwarding others' messages. In order to get credit for forwarding others' messages, a node needs to report to the CCS which messages it has helped to forward. In order to save storage, each mobile node should report to the CCS whenever it switches to a fast connection and has backup power. A mobile node can also use a desktop computer as a proxy to report to the CCS. In order to save bandwidth and storage, instead of requiring the whole message as a report, our system uses small *receipts*. Such receipts are derived from the content of the messages but do not expose the exact content of the messages. Thus, although we require that the CSS be trusted in terms of maintaining credit balance, the nodes do not need to trust the CSS in terms of message confidentiality.

### B. Who pays whom?

Before determining the amount of credit or charge to each node, we first discuss two basic questions.

The first question is who pays whom. Considering the relay of a message from a sender to a destination as a transaction, we need to decide who should be charged for the message and who should receive credit for relaying the message.

Although we can charge the destination, we decide that charging the sender will be a more robust and general approach. There are two reasons for charging the sender only. First, charging the destination may allow other nodes to launch a denial-of-service attack on the destination by sending it a large amount of traffic. Even sharing the cost between the sender and the destination could have a similar problem, because the sender could collude with the intermediate nodes, who could secretly return the sender's payment back, so that only the destination pays for the traffic. On the other hand, if only the sender is charged, a node will not have incentive to send useless messages. Second, if the destination benefits from the content of a message and thus should pay for it, the sender can get compensation from the destination, for example, through an application-layer payment protocol. Given these reasons, only the sender will be charged in our system.

A closely related question is who will receive credit for forwarding a message. Ideally, any node who has ever tried to forward a message should be compensated because forwarding a message will incur a cost to the node, no matter successful or not. However, a forwarded message may be corrupted on the link, and there is no way to verify that the forwarding action does occur. Although some local wireless networks such as IEEE 802.11 do provide link layer acknowledgments, such acknowledgment schemes are not universal and we refrain from changing basic network functions. Given this decision, the credit that a node receives will depend on whether or not its forwarding action is successful — a forwarding is successful if and only if the next node on the path receives the message. In other words, the CCS believes that a node has forwarded a message if and only if there is a successor of that node on the path reporting a valid receipt of the message.

### C. Objectives of the payment scheme

The second basic question is about the objective of the payment scheme. After all, the objectives of our payment scheme are to prevent cheating actions and to provide incentive for the nodes to cooperate. Given such objectives, our system does not target *balanced payment*; that is, we do not require that the total charge to the sender be equal to the total credit received by other nodes for a message. In fact, in order to prevent one type of cheating actions, our CCS charges the sender more than it gives to the other nodes (see Section III-F). In order to offset long-term net outflow of credit from the mobile nodes to the CCS, if in a large network, the CCS periodically returns the credit back to the mobile nodes uniformly; otherwise, the CCS periodically gives each mobile node a fixed amount of credit. Note that this return will not enable any cheating action or reduce the incentive of the nodes to forward others' messages.

*D. Cheating actions in the receipt-submission game*

Since the mobile nodes are selfish, without a proper payment scheme, they may not forward others' messages or they may try to cheat the system, if the cheating can maximize their welfare. In particular, a selfish node can exhibit one of the three selfish actions:

1) After receiving a message, the node saves a receipt but does not forward the message;
2) The node has received a message but does not report the receipt;
3) The node does not receive a message but falsely claims that it has received the message.

Note that any of the selfish actions above can be further complicated by collusion of two or more nodes. We next progressively determine the requirements on our system in order to prevent the above actions.

*E. Motivating nodes to forward messages*

In order to motivate a selfish node to forward others' messages, the CCS should give more credit to a node who forwards a message than to a node who does not forward a message. A basic scheme to achieve this objective is as follows. First, the CCS determines the last node on the path that has ever received the message. Then the CCS asks the sender to pay $\beta$ to this node, and $\alpha$ to each of its predecessors, where $\beta < \alpha$. Note that the CCS does not ask the sender to pay anything to the successors of the last node. Comparing this scheme with those in [4] and [5], we observe that the approaches in [4] and [5] are just the special case that $\beta$ is very small and $\alpha$ is close to 1. Figure 2 illustrates the basic idea with an example. In this example, only the first three intermediate nodes submit their receipts. Therefore, nodes 1 and 2 will each receive a payment of $\alpha$, and node 3 a payment of $\beta$. Since node 4 and the destination do not submit any receipt, they do not receive any credit. The sender pays a total of $2\alpha + \beta$.

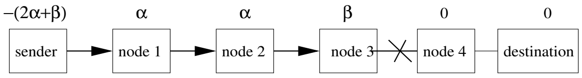


Fig. 2. Illustration of our payment scheme (version 1).

*F. Motivating nodes to report their receipts*

Obviously, each single node having received a message is motivated to report its receipt, if $\beta$ is greater than its cost of submitting a receipt, which, as we discussed previously, should be low since a receipt is generally small.

Unfortunately, there is still a collusion that can work against the above design. As an example, the last node (or in the general case, the last $k$ nodes) ever received the message can collude with the sender. In particular, if the last node does not report its receipt, the sender saves $\alpha$ while the last node loses $\beta$. However, if the sender gives the last node a behind-the-scene compensation of $\beta + \epsilon$, where $\epsilon > 0$, the last node will be better-off while the sender still enjoys a net gain of $\alpha - (\beta + \epsilon)$. Thus, the colluding group gets a net benefit of $\alpha - \beta$.

In order to prevent this cheating action, the CCS charges the sender an extra amount of credit if the destination does not report the receipt for message. This extra charge goes to the CCS instead of any nodes. The overall charge to the sender (including payments to other nodes and the extra charge) should be $k\beta$ less than the charge to the sender when the message arrives at the destination, where $k$ is the number of nodes not submitting receipts. Given such extra charge, even a colluding group cannot benefit from this cheating action. Again consider the example in Figure 2. Figure 3 shows the revised amount paid by the sender, which is equal to $(4\alpha + \beta) - 2\beta)$.

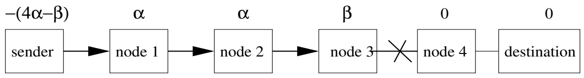


Fig. 3. Illustration of our payment scheme (version 2).

*G. Preventing false receipts*

Next we consider a countermeasure to the third type of selfish actions. As we discussed before, in order to save bandwidth and storage, our system requires that the nodes submit receipts instead of full messages. Given such a scheme for receipts, a group of colluding nodes can try to attack our system in several ways. For example, instead of forwarding the whole message, an intermediate node can forward only the receipt of a message to its successor, which is sufficient for getting credit. Moreover, the intermediate node can even wait until it has a fast connection to the successor to forward the false receipt, thus further saving resource usage.

The key to prevent such attack depends on the destination. We distinguish two cases: 1) the destination colludes with the intermediate nodes; or 2) the destination does not collude with the intermediate nodes.

We first consider the case that the destination colludes with the intermediate nodes, and therefore submits a receipt of a message even when it does not receive the whole message. For this case, we argue that the intermediate nodes and the destination should be paid as if no cheating had happened, because after all, the message is for the destination and the destination does submit a receipt for the message, indicating that it has received the message.

If the sender needs to make sure that the destination receives the whole message, a higher-layer protocol to validate the receipt of the whole message by the destination can be easily implemented, *e.g.*, see [28].

We next consider the case that the destination does not collude with the intermediate nodes. In this case, if the intermediate nodes forward only the receipt of a message instead of the whole message, then the destination will not be able to receive a valid message payload, and therefore will not submit a receipt for the message. Based on this observation, we can prevent the potential cheating action of the intermediate nodes by greatly reducing the amount of credit given to the intermediate nodes, if the message is not reported by the destination. With such reduction of credit, the cheating nodes cannot get enough credit even to cover the minimum expense needed for this type of cheating, *i.e.*, the cost of forwarding a receipt. To be more exact, if the destination does not report a receipt of a message, we multiply the credit paid to each node by $\gamma$, where $\gamma < 1$ (the exact requirement on $\gamma$ will be presented in Section V). Still consider the example in Figure 2. Figure 4 shows the revised amount of credit received by each node. In particular, comparing Figure 4 with Figure 3, due to this revision, we reduce the charge to the sender by $\gamma\beta$ instead of $\beta$, for each node on the path who does not report a receipt.

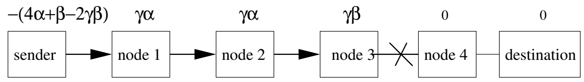


Fig. 4. Illustration of our payment scheme (final version).

## IV. Message-Forwarding Protocol: Specification

In the following formal specification of our protocol, we denote the public/private key pair of node $n_i$ by $(PK_i, SK_i)$. Each node $n_i$ maintains a sequence-number matrix $seq_i$, where $seq_i(j,k)$ is the sequence number of messages from sender $n_j$ to destination $n_k$, observed by node $n_i$. We assume that $(sign_{SK}(), verify_{PK}())$ is a digital signature scheme. In practice, we can use the RSA or the elliptic curve signature scheme.

### A. Sending a message

Suppose that node $n_0$ is to send message payload $m$ with sequence number $seq_0(0,d)$ to destination $n_d$, through path $p$. Node $n_0$ first computes a signature, $s$, on $(MD(m), p, seq_0(0,d))$, where $MD()$ is a message digest function such as MD5 [26] or SHA-1 [8]. Then, $n_0$ transfers $(m, p, seq_0(0,d), s)$ to the next hop and increases $seq_0(0,d)$ by 1. Figure 5 specifies the complete protocol steps.

$\triangleright$ $m$ is the message payload.
$\triangleright$ $n_0$ is the sender, $n_d$ the destination, and $p$ the path.

$s \leftarrow sign_{SK_0}(MD(m), p, seq_0(0,d))$
send $(m, p, seq_0(0,d), s)$ to the next node
$seq_0(0,d)++$

Fig. 5. Node $n_0$ sends a message to $n_d$.

### B. Receiving a message

Suppose that node $n_i$ receives $(m, p, seq, s)$. It first checks three conditions: 1) $n_i$ is on the path; 2) the message has a sequence number greater than $seq_i(0,d)$; and 3) the signature is valid. If any of the conditions is not satisfied, the message is dropped. Otherwise, it saves $(MD(m), p, seq, s)$ as a receipt. If $n_i$ is not the destination and decides to forward the message, it sends $(m, p, seq, s)$ to the next hop. Figure 6 specifies the protocol steps.

$\triangleright$ $(m, p, seq, s)$ is the received message.
$\triangleright$ $n_0$ is the sender, $n_d$ the destination.

**if** (($n_i$ not in $p$) || ($seq \leq seq_i(0,d)$)
  || ($verify_{PK_0}((MD(m), p, seq), s) \neq TRUE$))
  drop the message
**else**
  $seq_i(0,d) \leftarrow seq$
  save $(MD(m), p, seq, s)$ as a receipt
  **if** ($n_i$ is not the destination **and** decides to forward)
    send $(m, p, seq, s)$ to next hop
  **else**
    drop the message

Fig. 6. Node $n_i$ receives $(m, p, seq, s)$.

### C. Computing payments

A receipt $(D, p, seq, s)$ submitted by node $n_i$ is regarded as valid if

$$verify_{PK_0}((D, p, seq), s) = TRUE,$$

where $PK_0$ is the public key of the sender.

Without loss of generality, we assume that $p = (n_0, n_1, \ldots, n_e, \ldots, n_d)$, where $n_e$ is the last node on path $p$ that submits a valid receipt with sequence number $seq$. Then the CCS charges $C$ from node $n_0$, and pays $P_i$ to node $n_i$, where

$$C = (d-1)\alpha + \beta - (d-e)\gamma\beta,$$

$$P_i = \begin{cases} \alpha & \text{if } i < e = d \\ \beta & \text{if } i = e = d \\ \gamma\alpha & \text{if } i < e < d \\ \gamma\beta & \text{if } i = e < d. \end{cases}$$

Note that in implementation, the CCS will issue credit gradually. For example, when the last intermediate node submits its receipt for a message but the destination has not submitted its receipt yet, the last intermediate node will get $\gamma\beta$. Later, when the destination submits its receipt, the node will get its full credit of $\alpha$.

## V. MESSAGE-FORWARDING PROTOCOL: A FORMAL MODEL AND ANALYSIS

### A. A model of the receipt-submission game

For convenience of analysis, we model the submissions of receipts regarding a given message $m$ as a one-round game.

**Players.** This game has $d + 1$ players, $n_0, n_1, \ldots, n_d$, from the sender to the destination.[3]

**Players' Information.** Let $T_i$ be the information held by player $n_i$ that is unknown to the CCS. For $i > 0$, $T_i = TRUE$ if node $n_i$ has ever received message $m$; $T_i = FALSE$ otherwise. Obviously, the sender $n_0$ and the set of nodes that have ever received message $m$ constitute a prefix of the path. Therefore,

$$T_i = \begin{cases} TRUE & \text{if } 0 < i \le e' \\ FALSE & \text{if } e' < i \le d, \end{cases}$$

where $e'$ is the index of the last node that has ever received message $m$. Note that $e'$ is secret to the CCS when the game starts. Also note that a player has some partial information about $e'$, *i.e.*, the information inferred from its own information. For completeness, we define $T_0 = TRUE$.

**Actions.** Each player, $n_i$ ($i > 0$), has two possible actions: reporting that it has ever received message $m$ (by submitting a valid receipt), or withholding its report. We denote the action of $n_i$ by $A_i$. Then $A_i$ is either $TRUE$ or $FALSE$. The only exception is $n_0$, which has no choice of action. We define $A_0 = TRUE$, for completeness of our model.

**Cost of Actions.** We denote the cost of $n_i$'s action by $U_i$. As discussed before, in general, the cost of sending a receipt to the CCS is very low. However, if player $n_i$ does not receive message $m$ but can successfully claim that it has received the message, then a colluding node must have forwarded $n_i$ a copy of the receipt. Let $\delta$ be the cost of forwarding a receipt from one mobile node to another node. Then the colluding node incurs a cost of $\delta$ and $n_i$ must compensate the colluding node with $\delta$. Counting this cost on $n_i$, we have

$$U_i = \begin{cases} \delta & \text{if } T_i = FALSE \text{ and } A_i = TRUE \\ 0 & \text{otherwise.} \end{cases}$$

**Payment.** Recall that the system's payment to $n_i$ ($i > 0$) is

$$P_i = \begin{cases} \alpha & \text{if } i < e = d \\ \beta & \text{if } i = e = d \\ \gamma\alpha & \text{if } i < e < d \\ \gamma\beta & \text{if } i = e < d. \end{cases}$$

For $n_0$, the charge of $C$ can be viewed as a negative payment

$$P_0 = -C = -((d-1)\alpha + \beta - (d-e)\gamma\beta).$$

**Welfare.** For player $n_i$, deducting its cost from its received payment, the node has a welfare of

$$W_i = P_i - U_i.$$

### B. Analysis of the receipt-submission game: the security perspective

If $A_i = T_i$, then we say that $n_i$ *tells the truth*. Otherwise, we say that $n_i$ *cheats*. The *strategy* of $n_i$ can be *truth-telling*, *cheating*, or a probability distribution over these two choices. The *strategy profile* of a group of players refers to the ordered set of the strategies of these players.

*Definition 1:* For a player, an *optimal strategy* is a strategy that brings the maximum expected welfare to it, *regardless of the strategies of all the other nodes.*

*Theorem 1:* In the receipt-submission game, truth-telling is an optimal strategy for every node $n_i$, if $\delta \ge \gamma\beta$, and $n_d$ does not cheat in case of $T_d = FALSE$.
(Please see Appendix IX-A for a proof.)

Besides individual cheating, we further consider the possibility of collusion.

*Definition 2:* A game is *collusion-resistant*, if any group of colluding players cannot increase the expected sum of their welfare by using any strategy profile other than that in which everybody tells the truth.

*Theorem 2:* The receipt-submission game is collusion-resistant, if $\delta \ge (d-1)\gamma\alpha$, and $n_d$ does not cheat in case of $T_d = FALSE$.
(Please see Appendix IX-B for a proof.)

*Definition 3:* A game is cheat-proof, if truth-telling is an optimal strategy for every player and the game is collusion-resistant.

*Theorem 3:* The receipt-submission game is cheat-proof.

[3]Recall that each receipt contains the signed path. Therefore, nodes not on the path are easily excluded from this game.

### C. Analysis of performance: the incentive perspective

In the above proofs, we have essentially shown that each selfish node should report faithfully to the CCS. With this knowledge in mind, comparing the expected gain of credit from forwarding a message with that of not forwarding the message, an intermediate node can expect a net gain of $p_2\alpha+(p_1-p_2)\gamma\alpha+(1-p_1)\gamma\beta-\gamma\beta$, where $p_1$ is the probability that the message arrives at the next node, and $p_2$ the probability that the message arrives at the destination. Simplifying, we have $p_2(1-\gamma)\alpha+p_1\gamma(\alpha-\beta)$. Note that this payment gain is alway greater than 0 since $\gamma$ is small, and $\alpha > \beta$.

If this payment gain is sufficient to cover the cost of forwarding a message, the node has incentive to forward the message. Note that we can further fine-tune the payment parameters to optimize the system performance. However, this optimization issue is orthogonal to the main theme of this paper, and a thorough investigation of the optimization issue will be presented in a separate paper.

## VI. Stimulating Cooperation in Route Discovery and Multicast

Since route discovery uses broadcasts, the approach we have presented cannot be applied directly. Here we propose a slightly different approach, which is a bit more expensive. But since route discovery is performed less frequently, this approach is affordable in general. This approach is based on DSR, and essentially we will show how to improve DSR to stimulate cooperation in route discovery. Note that the reply to ROUTE REQUEST can be sent as a regular message. Therefore we only need to stimulate the re-broadcasting of ROUTE REQUEST.

### A. Sending a ROUTE REQUEST

In general, when a node starts to broadcast a ROUTE REQUEST, the message includes the source address and a sequence number. Then the node signs and broadcasts the message, and increases its sequence number counter by 1.

### B. Receiving a ROUTE REQUEST

Suppose that a node receives a ROUTE REQUEST. It first decides whether the message is a replay by looking at the sequence number. The node saves the received ROUTE REQUEST for getting payment in the future. When the node decides to rebroadcast the ROUTE REQUEST, it appends its own address to the ROUTE REQUEST and signs the extended message.

### C. Computing payment

When the CCS computes payment, a ROUTE REQUEST is rejected if any signature in the message is invalid. Furthermore, if a ROUTE REQUEST submitted by a node is a part of another ROUTE REQUEST submitted by the same node, then the former message is rejected. Finally, the CCS builds a tree based on the accepted ROUTE REQUEST messages. The sender pays $\alpha$ to each non-leaf node of the tree, and $\beta$ to each leaf of the tree. For each node outside the tree, the sender node pays $\alpha-\beta$ to the CCS.

### D. Discussion and extension

The above approach is provably secure for route discovery. We omit the proof since it is similar to the proofs we presented for stimulating forwarding data messages in unicast. As route-discovery broadcast can be viewed as a special case of multicast, this approach can also be applied to multicast if multicast is not frequently used in the system. If multicast is frequently used, we can use a combination of the above approach and the approach presented for stimulating forwarding messages, which is less expensive. However, we do not have a provable result for the second type of cheating in this case. We leave the proof as a future research topic.

## VII. Evaluations

### A. Overhead

We first evaluate the overhead of our system. In order to measure the overhead, we have implemented a prototype of our system using the Crypto++4.0 library [7]. The implementation can run over a wide range of platforms such as Linux and Win32.

In the evaluations below, our mobile node is a Laptop with an *Intel Mobile Pentium III* processor at 866MHz. The OS of the mobile node is Windows XP. The length of a message payload is 1000 bytes. The message digest function is MD5. We consider two digital signature schemes: RSA with a modulus of 1024 bits, and ECNR over GF(p) 168 [12]. We assume that the average path length is 8 hops.

We first evaluate the CPU processing time on a mobile node. In our system, the major online processing overhead is the signing operation by the sender and the verification operation by the intermediate nodes. The second and third columns of Table I show the CPU processing time of the sender to send a message and that of an intermediate node to forward a message, respectively. We observe that RSA has a much smaller forwarding overhead. Thus, if reducing forwarding overhead is the major objective, RSA is a better implementation choice. However, for

| | send (ms) | forward (ms) | authentication header (bytes) | receipt (bytes) |
|---|---|---|---|---|
| RSA 1024 | 10.4 | 0.3 | 128 | 180 |
| ECNR over GF(p) 168 | 7.3 | 13.2 | 42 | 94 |
| ECNR over GF(p) 168 (precomputation) | 3.7 | 6.1 | 42 | 94 |

TABLE I
CPU PROCESSING TIME; SIZES OF AUTHENTICATION HEADER AND RECEIPTS.

both schemes, we observe that the CPU processing time is acceptable, if the nodes do not send a large number of messages, which is the expected case when the mobile nodes have limited bandwidth and energy.

We next evaluate the bandwidth and storage requirement. Compared with a message using DSR as the routing protocol but without message authentication, the major increased overhead is the digital signature for message authentication. For RSA with a modulus of 1024 bits, the authentication header is about 128 bytes; for ECNR GF(p) with 168 bits, the header is about 42 bytes. In terms of storage requirement for the receipts, for RSA 1024, the total storage of a receipt is 180 bytes, and for the Elliptic Curve based ECNR, it is 94 bytes. Comparing RSA with ECNR, we observe that ECNR has a much smaller bandwidth and storage requirement.

### B. System performance vs. network resource

We next evaluate the performance of our system. One major metric of the performance of our system is the message success rate, *i.e.*, the percentage of messages that are successfully relayed from the sender to the destination. For the purpose of this evaluation, we ignore message drops due to channel errors.

We first note that this success rate will depend on the sending/forwarding strategy of the mobile nodes. As we have discussed in Section III, although our system provides incentive for cooperation by giving more credit for forwarding a message, whether or not to forward a specific message will depend on the objectives and the status of a node. To demonstrate the generality of our system, for the purpose of this evaluation, we consider a special class of mobile nodes, namely the power-and-credit-conservative nodes. Specifically, a node is power-conservative if its remaining power allows it to send (and forward) only a limited amount of messages; a node is credit-conservative if it refrains from sending any new message when its credit balance is insufficient to cover the charge for sending a message. For this type of nodes, we can show that, if the objective of such a node is to maximize the total number of its own messages sent and at the same time to send the messages as early as possible, then the optimal send/forward strategy is the following: when it receives a transient message, if the number of messages allowed to be sent by its estimated credit balance is smaller than the number of messages allowed to be sent by its remaining battery, forward the transient message and increase its estimated credit balance by $p\alpha$, where $p$ is the probability that the forwarded message will arrive at its destination; otherwise, drop the message. In summary, let $c$ and $b$ denote the estimated credit balance and the number of messages allowed to be transmitted by the remaining battery of a node, respectively. Assume that each message takes an average of $L$ hops. Then the policy of such a node is the following: if $\frac{c}{L} < b$, forward a transient message; otherwise, drop the message. Given the strategy above, we next evaluate the message success rate of our system.

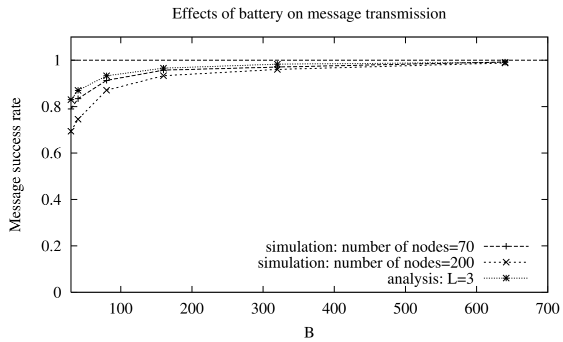


Fig. 7. Message success rate vs. network battery resource.

We first evaluate the message success rate under different configurations of network resource. Figure 7 shows the message success rates for two ad hoc networks: one network with 70 nodes uniformly distributed in an area of 1000 by 1000, and another network with 200 nodes uniformly distributed in an area of 2000 by 2000. The communication radius of each node is 250. In this experiment, since the nodes are power-and-credit-conservative, their estimated credit balance $c$ is close to 0 and we choose their initial credit to be uniformly distributed in $[0, C]$, where $C = 10$. To observe the effect of the amount of node resource on the overall message success rate, for each node, we choose its $b$, the number of messages that can be sent/forwarded by the remaining battery of the node, uniformly from $[0, B]$, where $B$ is from 30 to 640. Note that even the maximum number of 640 is very conservative [32]. For this scenario, first we can drive an approximate analytical expression for the message success rate as

$(1-\frac{C+1}{2BL})^L$, where $L$ is the average path hops. In addition to this analytical result, Figure 7 also plots the results from simulations in order to capture the full details such as traffic concentration. To control the number of experiments for each configuration, we repeat the experiment of a configuration with a different random seed until the 5% confidence interval is narrower than 5% of the mean value. From Figure 7, we observe clearly that with increasing resource, the nodes are more willing to forward others' messages, and therefore the message success rate is very close to 1.

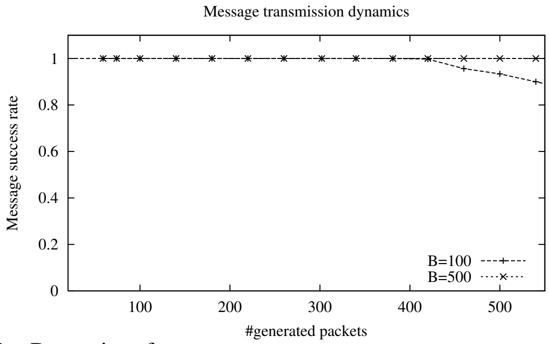


Fig. 8. Dynamics of message success rate.

We next evaluate the dynamics of message success rate; that is, how message success rate evolves as the nodes send more messages. Figure 8 shows the result. Under this experiment, the initial credit of each node is 3, and the initial battery of each node is $B$, where $B = 100$ or $500$. The value of $B = 100$ is in the very low end, and the objective is to observe message drops. The x-axis of Figure 8 is the index of the number of messages generated by the mobile nodes, and the y-axis shows the message success rate. From this figure, we observe that as system evolves and no new node joins, the batteries of the nodes are consumed and the nodes tend to be more conservative. However, we observe that, even in a low battery configuration, considerable number of messages will be generated before the message success rate decreases.

## VIII. Conclusion

In this paper, we presented Sprite, a system to provide incentive to mobile nodes to cooperate. Our system determines payments and charges from a game-theoretic perspective, and we showed that our system motivates each node to report its behavior honestly, even when a collection of the selfish nodes collude. We also modeled the essential component of our system as the receipt-submission game, and proved the correctness of our system under this model. As far as we know, this is the first pure-software solution that has formal proofs of security. Our main result works for packet-forwarding in unicast, and we extended it for route discovery and multicast as well. We also implemented a prototype of our system and showed the overhead of our system is insignificant. Simulations and analysis of the power-and-credit-conservative nodes showed that the nodes can cooperate and forward each other's messages, unless the resource of the nodes is extremely low.

## References

[1] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J. P. Hubaux, and J. Y. Le Boudec. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 39(6), June 2001.

[2] Sonja Buchegger and Jean-Yves Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002.

[3] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes - fairness in dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*. IEEE, June 2002.

[4] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, August 2000.

[5] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, summer 2002.

[6] Bridget Dahill, Brian Neil Levine, Elizabeth Royer, and Clay Shields. A secure routing protocol for ad hoc networks. Technical report, UMass, 2001.

[7] Wei Dai. Crypto++4.0. Available at http://www.eskimo.com/ weidai/cryptlib.html.

[8] Federal Information Processing Standards Publication 180-1. *Secure hash standard*, 1995.

[9] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing.*, 2002.

[10] Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences (Special issue on Internet Algorithms.)*, 63:21–41, 2001.

[11] Philippe Golle, Kevin Leyton-Brown, and Ilya Mironov. Incentives in peer-to-peer file sharing. In *Proceedings of the ACM Symposium on Electronic Commerce (EC' 01) 2001*, Tampa, FL, October 2001.

[12] IEEE P1363 Group. IEEE P1363 standard. Available at http://grouper.ieee.org/groups/1363/index.html.

[13] John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth. In *Proceedings of the 42th IEEE Symposium on Foundations of Computer Science*, pages 252–259, 2001.

[14] Hung-Yun Hsieh and Raghupathy Sivakumar. Performance comparison of cellular and multi-hop wireless networks: A quantitative study. In *Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2001*, Cambridge, MA, June 2001.

[15] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.

[16] J. P. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, October 2001.

[17] J. P. Hubaux, J. Y. Le Boudec, S. Giordano, and M. Hamdi. The terminodes project: Towards mobile ad-hoc wan. In *Proceedings of MOMUC'99*, San Diego, November 1999.

[18] D. B. Johnson and D. A. Malt. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, Chapter 5, (Tomasz Imielinski and Hank Korth, eds.). Kluwer Academic Publishers, 1996.

[19] Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-securing ad-hoc wireless networks. In *ISCC*, 2002.

[20] Sergio Marti, Thomas Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of The Sixth International Conference on Mobile Computing and Networking 2000*, Boston, MA, August 2000.

[21] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(166–196), 2001.

[22] Noam Nisan. Algorithms for selfish agents. In *16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 1–15, 1999.

[23] Martin J. Osborne and Ariel Rubenstein. *A Course in Game Theory*. The MIT Press, 1994.

[24] Christos H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33th Annual ACM Symposium on the Theory of Computing*, pages 749–753, 2001.

[25] Charles Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[26] R. Rivest. The MD5 message-digest algorithm. *RFC 1321*, 1992.

[27] Tim Roughgarden and Eva Tardos. How bad is selfish routing? *Journal of the Association for Computing Machinery*, 49(2):236–259, 2002.

[28] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. *ACM Computer Communication Review*, 29(5), October 1999.

[29] Akis Spyropoulos and Cauligi Raghavendra. Energy efficient communications in ad hoc networks using directional antennas. In *Proceedings of IEEE INFOCOM '02*, New York, NY, June 2002.

[30] Mark Stemm and Randy H. Katz. Vertical handoffs in wireless overlay networks. *Mobile Networks and Applications*, 3(4):335–350, 1998.

[31] M. Vetterli Th. Gross, J. P. Hubaux, and J. Y. Le Boudec. Towards self-organizing mobile ad-hoc networks: the terminodes project. *IEEE Comm Mag*, 39(1):118 –124, January 2001.

[32] C-K Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall PTR, 2001.

[33] Jeffrey E. Wieselthier, Gam Nguyen, and Anthony Ephremides. Energy-limited wireless networking with directional antennas: The case of session-based multicasting. In *Proceedings of IEEE INFOCOM '02*, New York, NY, June 2002.

[34] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 1999.


## IX. APPENDIX

### A. Proof of Theorem 1

*Proof:* Consider a strategy profile of all of the rest players, in which each player, $n_j$ ($j \neq i$), tells the truth with probability $x_j$. We distinguish four cases here.

- Case A. $i = 0$. Since $A_i = T_i = TRUE$ is the only possible strategy, it is also the best response.
- Case B. $0 < i < e'$. Recall that $e'$ is the index of the last node that has ever received the message. If $n_i$ tells the truth, its expected welfare $EW_i^+ = EP_i^+$ is

$$\begin{cases} (1 - \prod_{j=i+1}^{e'}(1-x_j)\prod_{j=e'+1}^{d-1} x_j)\gamma\alpha \\ + \prod_{j=i+1}^{e'}(1-x_j)\prod_{j=e'+1}^{d-1} x_j\gamma\beta & \text{if } e' < d \\ x_d\alpha + (1-x_d)((1-\prod_{j=i+1}^{e'-1}(1-x_j))\gamma\alpha \\ + \prod_{j=i+1}^{e'-1}(1-x_j)\gamma\beta) & \text{if } e' = d; \end{cases}$$

if $n_i$ cheats, its expected welfare $EW_i^- = EP_i^-$ is

$$\begin{cases} (1 - \prod_{j=i+1}^{e'}(1-x_j)\prod_{j=e'+1}^{d-1} x_j)\gamma\alpha & \text{if } e' < d \\ x_d\alpha + (1-x_d)(1-\prod_{j=i+1}^{e'-1}(1-x_j))\gamma\alpha & \text{if } e' = d. \end{cases}$$

Therefore, we always have $EW_i^+ \geq EW_i^-$, which implies that telling the truth with probability 1 will bring the maximum expected welfare to $n_i$.
- Case C. $i = e'$. If $n_i$ tells the truth, its expected welfare $EW_i^+ = EP_i^+$ is

$$\begin{cases} (1 - \prod_{j=i+1}^{d-1} x_j)\gamma\alpha + \prod_{j=i+1}^{d-1} x_j\gamma\beta & \text{if } e' < d \\ \beta & \text{if } e' = d; \end{cases}$$

if $n_i$ cheats, its expected welfare $EW_i^- = EP_i^-$ is

$$\begin{cases} (1 - \prod_{j=i+1}^{d-1} x_j)\gamma\alpha & \text{if } e' < d \\ 0 & \text{if } e' = d. \end{cases}$$

As in the previous case, we always have $EW_i^+ \geq EW_i^-$, which implies that telling the truth with probability 1 will bring the maximum expected welfare to $n_i$.
- Case D. $e' < i \leq d$. Note that $T_d = FALSE$ here, which implies that $n_i$ always tells the truth in case of $i = d$. So we only need to consider the case of $i < d$. If $n_i$ tells the truth, the expected welfare is

$$EW_i^+ = EP_i^+ = (1 - \prod_{j=i+1}^{d-1} x_j)\gamma\alpha.$$

If $n_i$ cheats, it gets an expected payment of

$$EP_i^- = (1 - \prod_{j=i+1}^{d-1} x_j)\gamma\alpha + \prod_{j=i+1}^{d-1} x_j\gamma\beta,$$

while its gets a cost of

$$U_i^- = \delta.$$

So its expected welfare is

$$EW_i^- = EP_i^- - U_i^- = (1 - \prod_{j=i+1}^{d-1} x_j)\gamma\alpha + \prod_{j=i+1}^{d-1} x_j\gamma\beta - \delta.$$

As in Cases B and C, we always have $EW_i^+ \geq EW_i^-$, which implies that telling the truth with probability 1 will bring the maximum expected welfare to $n_i$.

∎

### B. Proof of Theorem 2

*Proof:* Consider strategizing group $G$ that uses a strategy profile other than everybody telling the truth. Suppose that each player $n_i \in G$ tells the truth with probability $x_i$. The expected sum of welfares of $G$ is

$$EW_G = \sum_{L \subseteq G} \prod_{i \in L} (1 - x_i) \prod_{i \in G-L} x_i W_G(L),$$

where $W_G(L)$ denotes the sum of welfares of $G$ in case that the set of lying players is $L$. Our goal is to show

$$EW_G \leq W_G(\phi).$$

Obviously, we only need to prove

$$\forall L \subseteq G, W_G(L) \leq W_G(\phi).$$

We distinguish two cases here. (Hereafter, we use $\sharp G(u,v)$ to denote $|\{i | u \leq i \leq v \wedge n_i \in G\}|$.)

- Case A. $n_0 \notin G$. By considering the indices of players in $L$, we further distinguish three subcases.
  - Subcase A-A. $\forall n_i \in L$, $i < e'$. Then trivially we have
  $$W_G(L) = W_G(\phi).$$
  - Subcase A-B. $\forall n_i \in L$, $i \leq e'$, and $n_{e'} \in L$. Then $W_G(L)$ is equal to
  $$\begin{cases} W_G(\phi) - ((e'-1)\gamma\alpha + \gamma\beta)) \\ \qquad \text{if } e' < d \text{ and } \forall i \leq e', n_i \in L \\ W_G(\phi) - ((e'-1)\alpha + \beta) \\ \qquad \text{if } e' = d \text{ and } \forall i \leq e', n_i \in L \\ W_G(\phi) - (\sharp G(e,e)(\gamma\alpha - \gamma\beta) \\ \quad + \sharp G(e+1, e'-1)\gamma\alpha + \gamma\beta) \\ \qquad \text{if } e' < d \text{ and } \exists i \leq e', n_i \notin L \\ W_G(\phi) - (\sharp G(1, e-1)(1-\gamma)\alpha \\ \quad + \sharp G(e,e)(\alpha - \gamma\beta) + \sharp G(e+1, e'-1)\alpha + \beta) \\ \qquad \text{if } e' = d \text{ and } \exists i \leq e', n_i \notin L, \end{cases}$$
  where $e = \max_{i \leq e', n_i \notin L} i$. Therefore,
  $$W_G(L) \leq W_G(\phi).$$
  - Subcase A-C. $\exists n_i \in L$, $i > e'$. Then
  $$\begin{aligned} W_G(L) = & W_G(\phi) + \sharp G(e', e')(\gamma\alpha - \gamma\beta) \\ & + \sharp G(e'+1, e-1)\gamma\alpha + \gamma\beta - \delta, \end{aligned}$$
  where $e = \max_{e' < i < d, n_i \in L} i$. It's easy to see
  $$\begin{aligned} W_G(L) & \leq W_G(\phi) + (e - e')\gamma\alpha - \delta \\ & \leq W_G(\phi) + (d-1)\gamma\alpha - \delta \\ & \leq W_G(\phi). \end{aligned}$$
- Case B. $n_o \in G$. As in Case A, we further distinguish three subcases.
  - Subcase B-A. $\forall n_i \in L$, $i < e'$. Trivially, we have
  $$W_G(L) = W_G(\phi).$$
  - Subcase B-B. $\forall n_i \in L$, $i \leq e'$, and $n_{e'} \in L$. Then $W_G(L)$ is equal to
  $$\begin{cases} W_G(\phi) + e'\gamma\beta - ((e'-1)\gamma\alpha + \gamma\beta)) \\ \qquad \text{if } e' < d \text{ and } \forall i \leq e', n_i \in L \\ W_G(\phi) + e'\gamma\beta - ((e'-1)\alpha + \beta) \\ \qquad \text{if } e' = d \text{ and } \forall i \leq e', n_i \in L \\ W_G(\phi) + (e'-e)\gamma\beta - (\sharp G(e,e)(\gamma\alpha - \gamma\beta) \\ \quad + \sharp G(e+1, e'-1)\gamma\alpha + \gamma\beta) \\ \qquad \text{if } e' < d \text{ and } \exists i \leq e', n_i \notin L \\ W_G(\phi) + (e'-e)\gamma\beta - (\sharp G(1, e-1) \\ \quad \cdot(1-\gamma)\alpha + \sharp G(e,e)(\alpha - \gamma\beta) \\ \quad + \sharp G(e+1, e'-1)\alpha + \beta) \\ \qquad \text{if } e' = d \text{ and } \exists i \leq e', n_i \notin L, \end{cases}$$
  where $e = \max_{i \leq e', n_i \notin L} i$. Since $\gamma\beta < \beta < \alpha$ and $\gamma\beta < \gamma\alpha$, it is easy to see
  $$W_G(L) \leq W_G(\phi).$$
  - Subcase B-C. $\exists n_i \in L$, $i > e'$. Then
  $$\begin{aligned} W_G(L) = & W_G(\phi) - (e - e')\gamma\beta + \sharp G(e', e') \\ & \cdot(\gamma\alpha - \gamma\beta) + \sharp G(e'+1, e-1)\gamma\alpha \\ & + \gamma\beta - \delta, \end{aligned}$$
  where $e = \max_{e' < i < d, n_i \in L} i$. It's easy to see
  $$\begin{aligned} W_G(L) & \leq W_G(\phi) + (e - e')\gamma\alpha - \delta \\ & \leq W_G(\phi) + (d-1)\gamma\alpha - \delta \\ & \leq W_G(\phi). \end{aligned}$$

∎